Building a Rack-Scale Computer with P4 at the Core

Ryan Goodfellow
Engineer @ **Oxide Computer Company**

Outline

Oxide product context

- What is a rack scale computer?
- Where does P4 fit in?
- Why is network programmability important?

`x4c` compiler intro.

- Why a new compiler?
- Workflows with `x4c`: development, testing and CI.

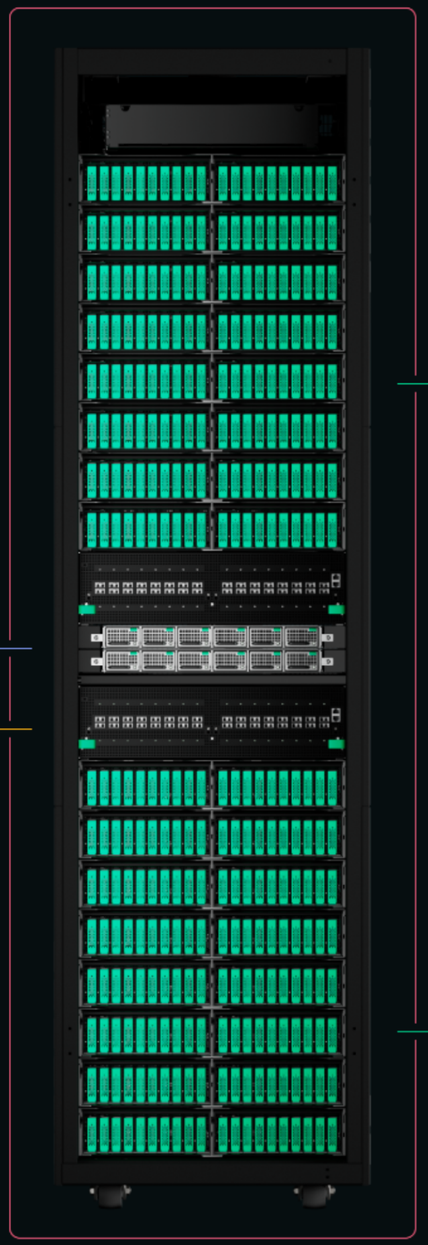
Live demos!

- Emulating an ASIC in a hypervisor.
- Writing networking tools in P4.

Hardware and software designed together

Vertically integrated and scale-ready. Bringing hyperscaler agility to the mainstream enterprise.

HOLISTIC SECURITY



```
resource
  "oxide_instance" {
    organization_name =
    project_name      =
    description       =
    name              =
    host_name         =
    memory            =
    ncpus             =
    attach_to_disks   =
    network_interface {
```

```
~> terraform init
Init modules...
Init backend...
Success

~> terraform appl
```

COMPUTE PROVISIONING SERVICES

HIGH EFFICIENCY POWER DESIGN

INTEGRATED NETWORK SERVICES

☐	Allow	TCP 80	ENABLED
☐	Allow	TCP 80	ENABLED
☑	Allow	ICMP	ENABLED
☐	Allow	TCP 0-65535	DISABLED
☐	Allow	TCP 3389	ENABLED
☐	Allow	TCP 22	DISABLED

VIRTUAL BLOCK STORAGE SERVICE

Disk-type

Balanced persistant disk

Source type

Contents of created disk

- ○ Blank disk
- ◉ Image
- ○ Snapshot

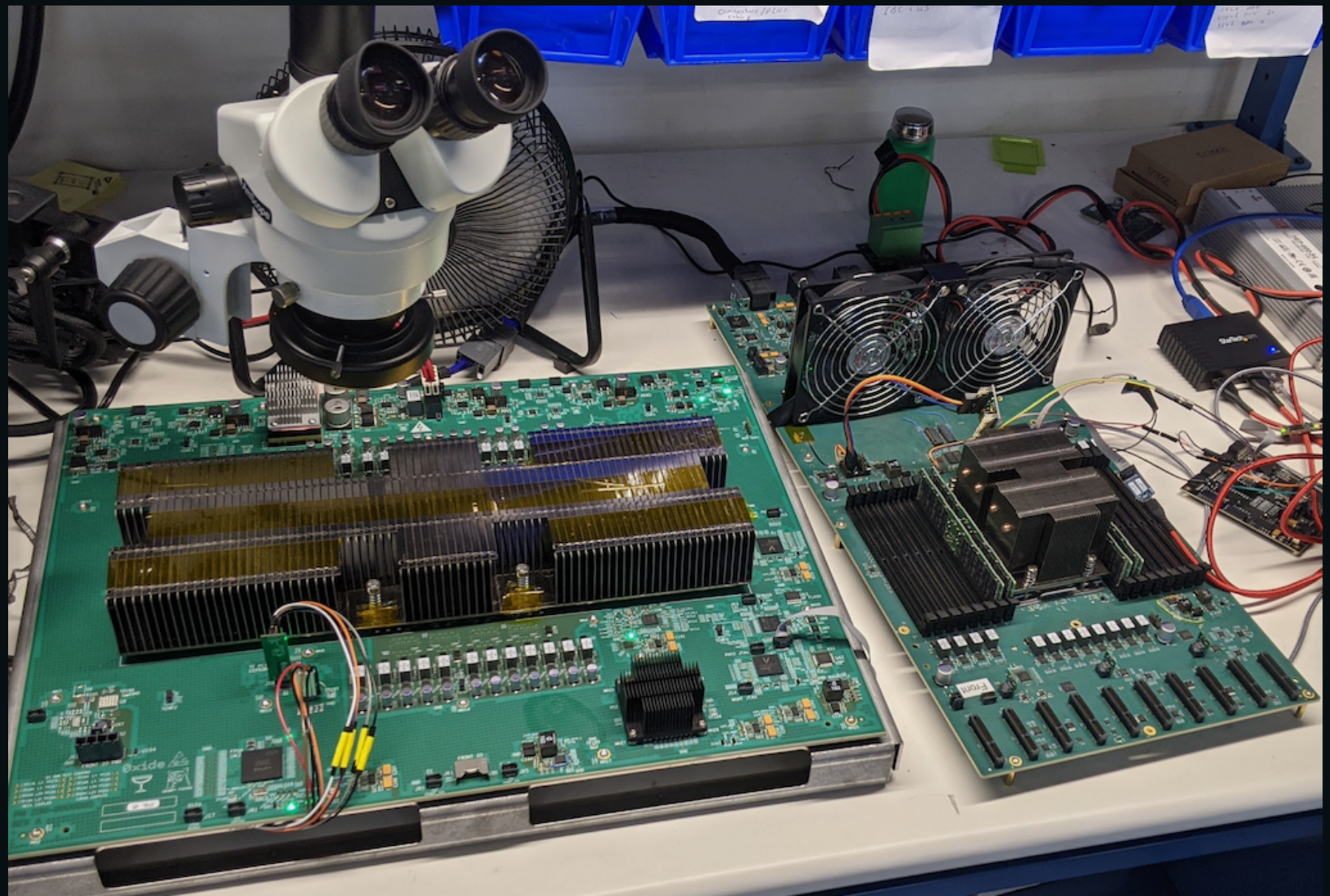
Source image

Disk size (GiB)

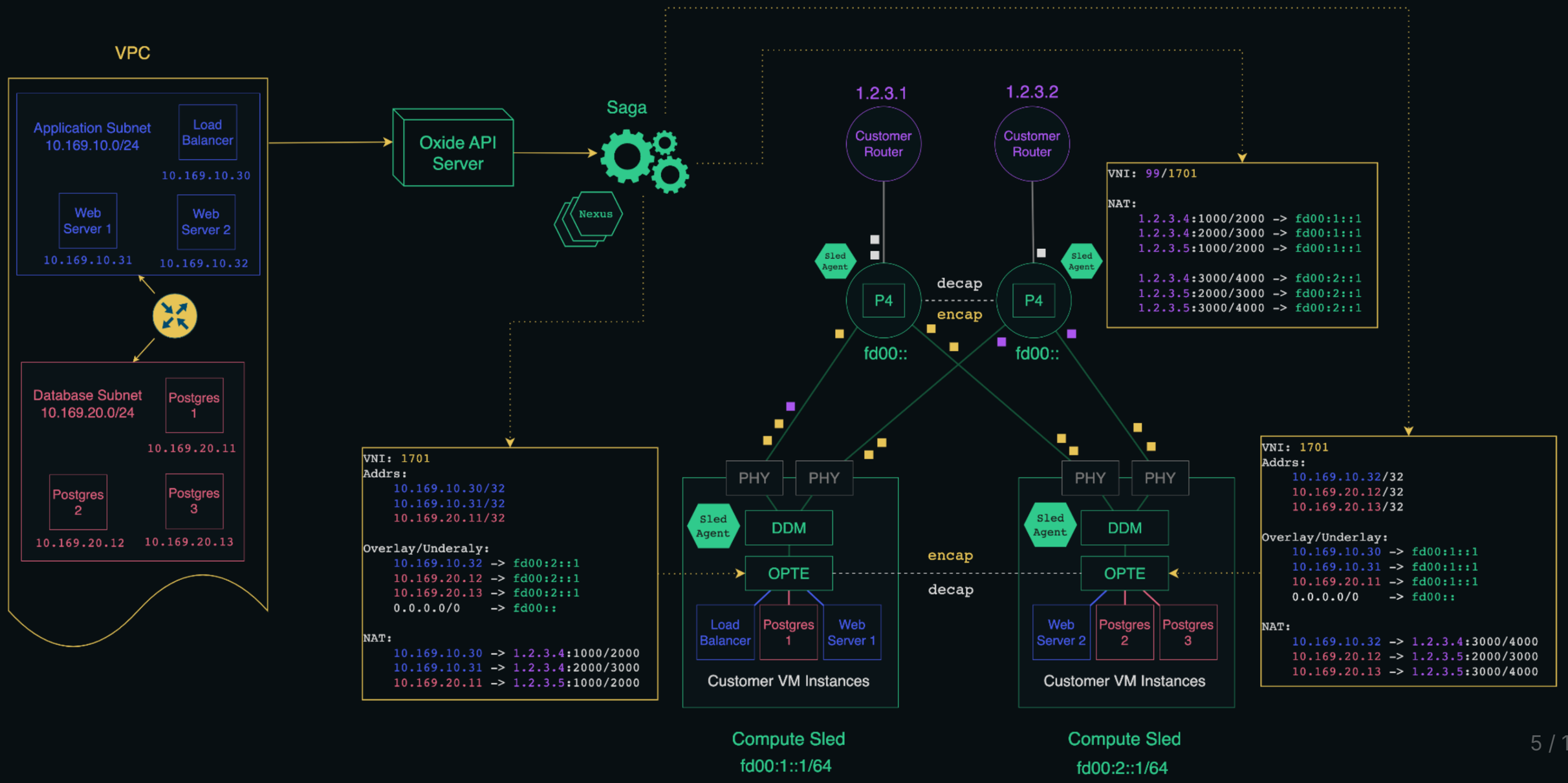
500

- Set as boot disk
- Detach from instance
- Clone disk
- Delete disk

Very custom hardware



Where does P4 fit in?



Why is network programmability important?

Flexibility

- Example: Cooperative distributed NAT.

Fit-for-purpose resource usage.

- Using precious TCAM for exactly what our requirements call for.

Comprehensibility

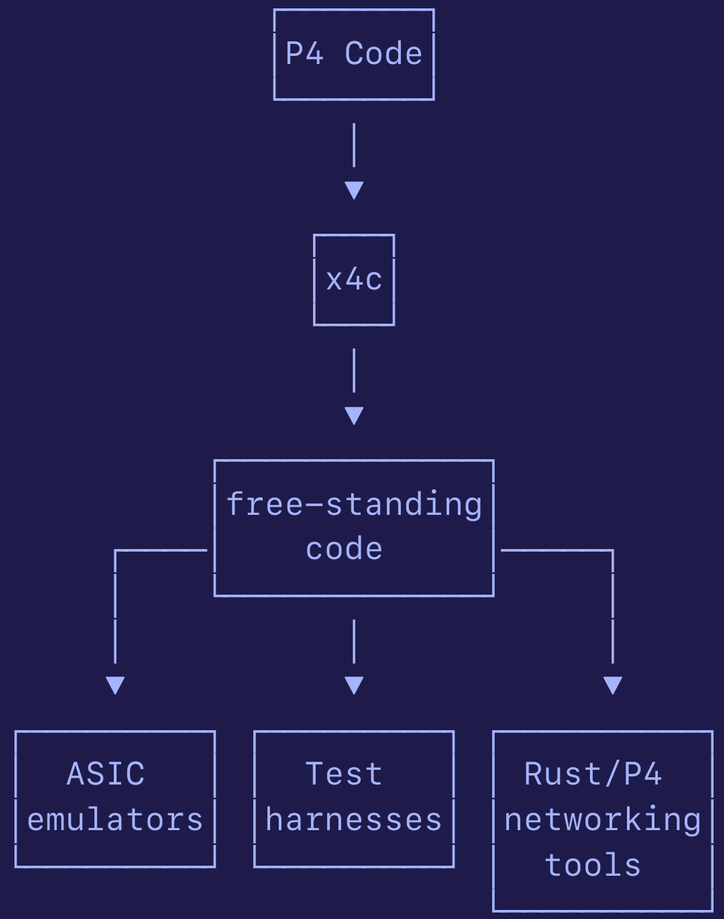
- Our customer operators have a responsibility to their organizations to keep mission-critical networks running.
- Giving them data plane code access gives them the autonomy comprehend network behaviors and solve problems independently.

Why a new compiler?

We needed a way to run P4 code in very different contexts.

In the kernel, in userpsace, embedded in Rust programs, free standing, in hypervisors, etc.

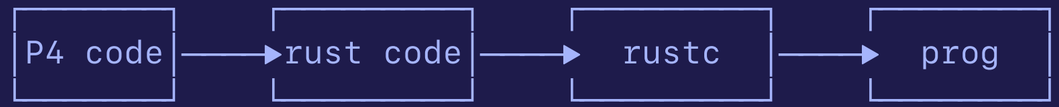
Compiling from P4 to Rust and then from Rust to whatever form we need makes a lot of sense.



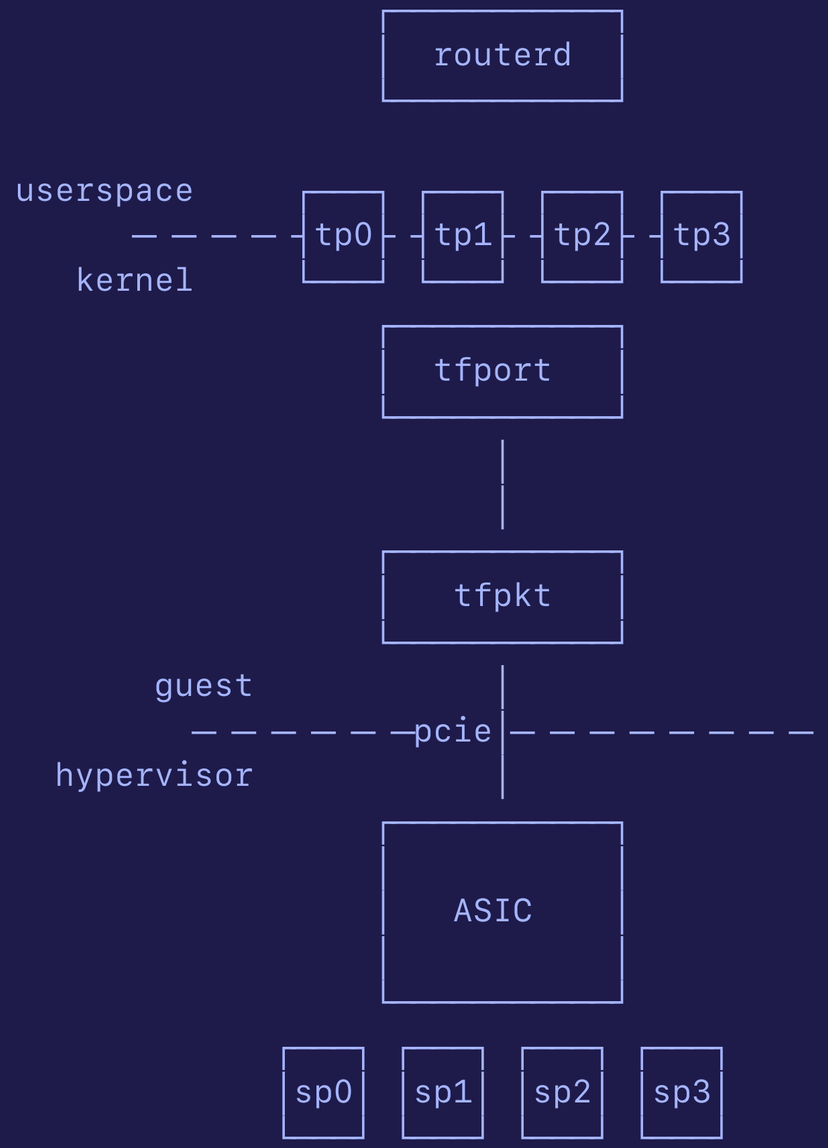
Workflows with x4c

Compilation to Pipeline Library

Direct Inclusion in Rust Code

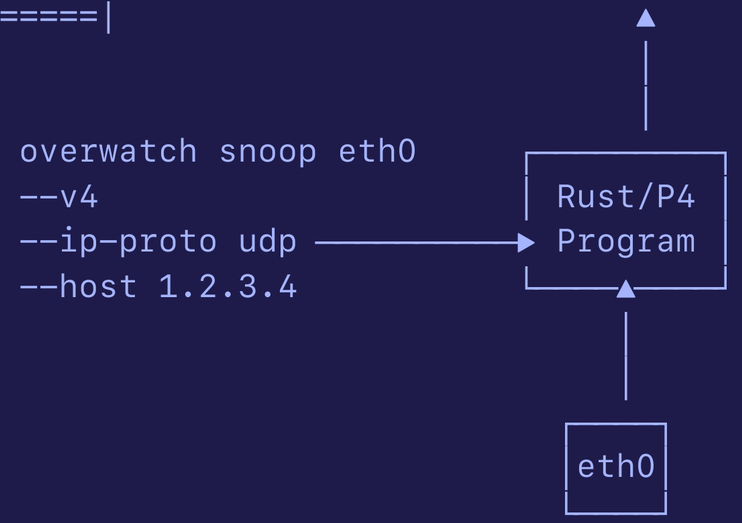


ASIC emulation in the Hypervisor



Writing networking tools in P4 and Rust

```
=====|
Eth  | 50:51:A9:FB:0A:37 > 01:00:5E:00:00:FB et IPv4 len 521
Ip4  | 192.168.1.107 > 224.0.0.251 ihl 5 ds 0 len 507 id 17527
     |     flags DF fo 0 ttl 1 chk 36972 proto Udp
UDP  | 5353 > 5353 len 487 chk 52730
=====|
Eth  | 50:51:A9:FB:0A:37 > 01:00:5E:00:00:FB et IPv4 len 521
Ip4  | 192.168.1.107 > 224.0.0.251 ihl 5 ds 0 len 507 id 17652
     |    flags DF fo 0 ttl 1 chk 36847 proto Udp
UDP  | 5353 > 5353 len 487 chk 52730
=====|
Eth  | 50:51:A9:FB:0A:37 > 01:00:5E:00:00:FB et IPv4 len 320
Ip4  | 192.168.1.107 > 224.0.0.251 ihl 5 ds 0 len 306 id 17868
     |    flags DF fo 0 ttl 1 chk 36832 proto Udp
UDP  | 5353 > 5353 len 286 chk 13449
=====|
```



Thank you!